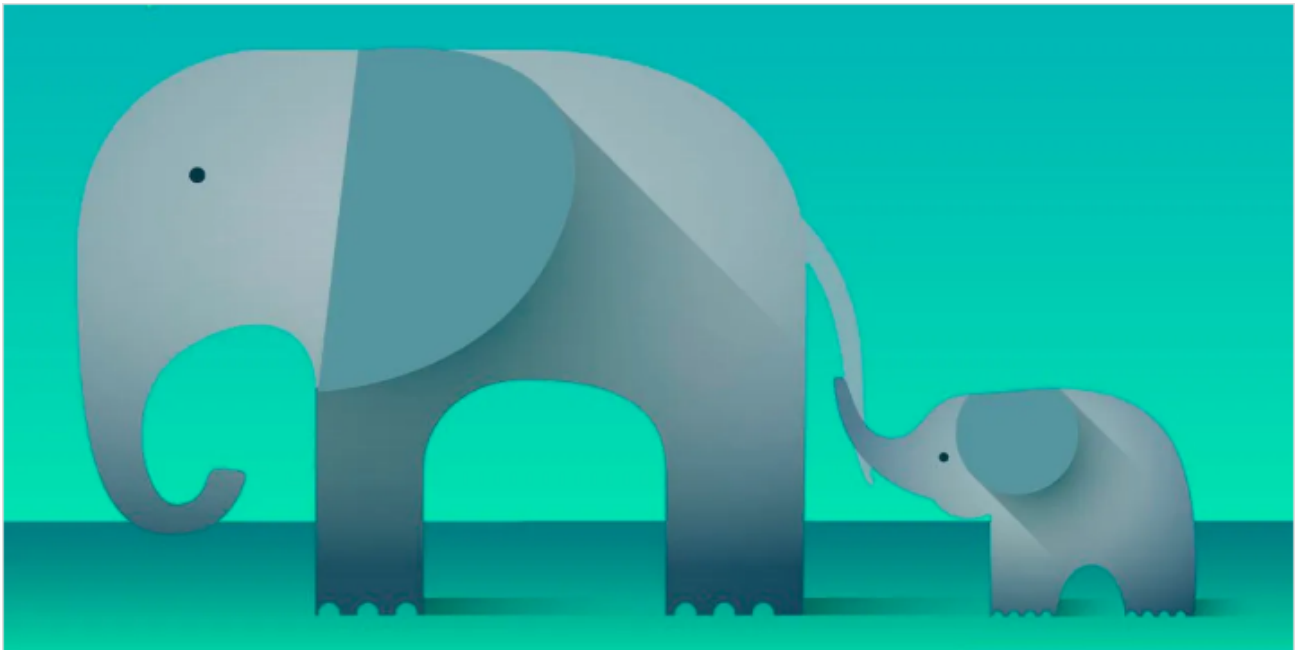
Documentation POO en Php

1-Ressources:

Pour la documentation j’ ai consulté plusieurs ressources principalement sur Youtube :



oop.php

Users > arajendran > Desktop > oop.php

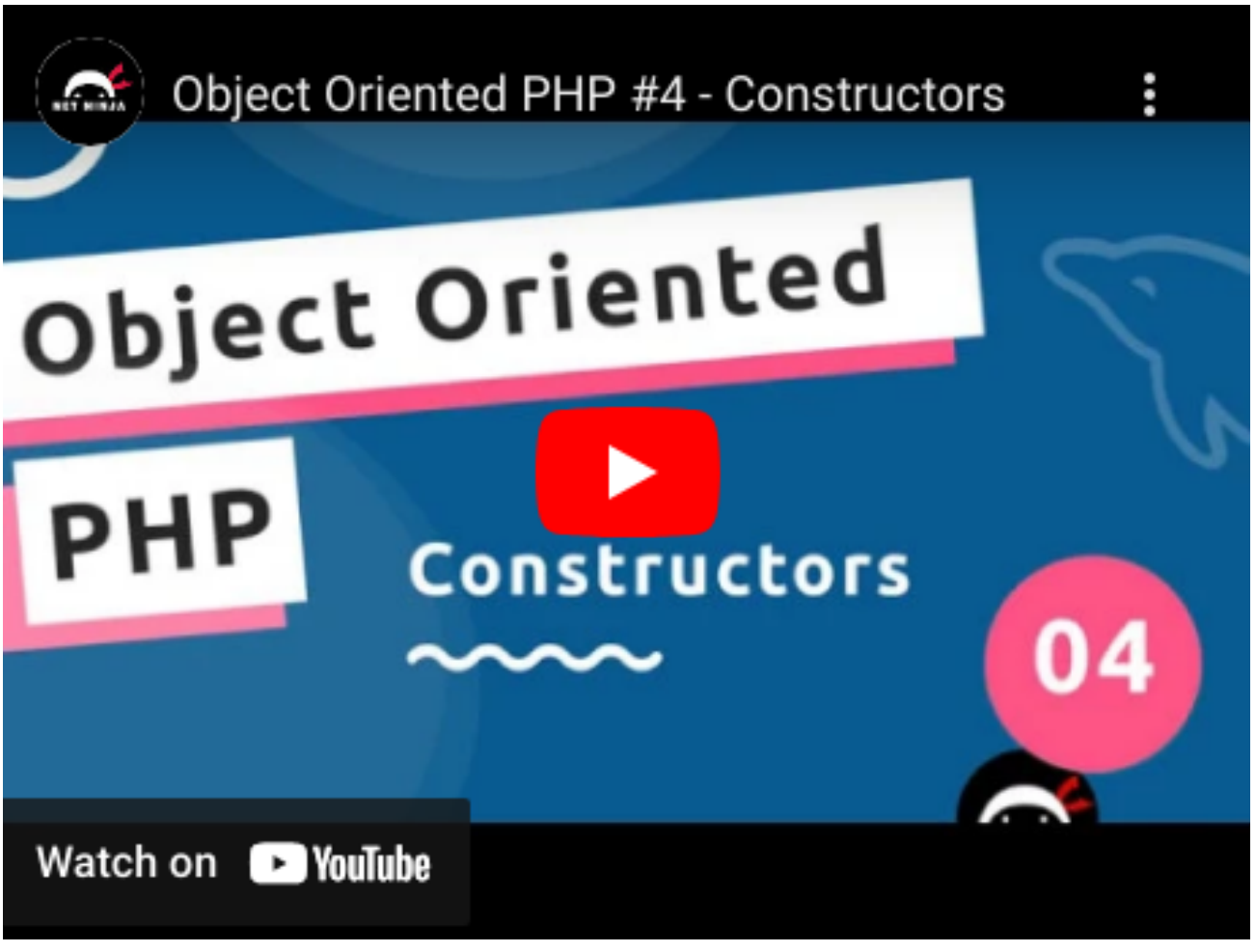
```php
<?php

class PaymentGateWay
{
    // Properties
    public $name;

    // Method to set name
    function setName($name)
    {
        $this->name = $name;
    }

    // Method to get name
    function getName()
    {
        return $this->name;
    }
}

$stripe = new PaymentGateWay();
$stripe->setName('Stripe');
print_r("Payment Gateway Name: " . $stripe->getName() . "\n");

$paypal = new PaymentGateWay();
$paypal->setName('Paypal');
print_r("Payment Gateway Name: " . $paypal->getName() . "\n");
?>
```

TERMINAL PROBLEMS

OUTPUT

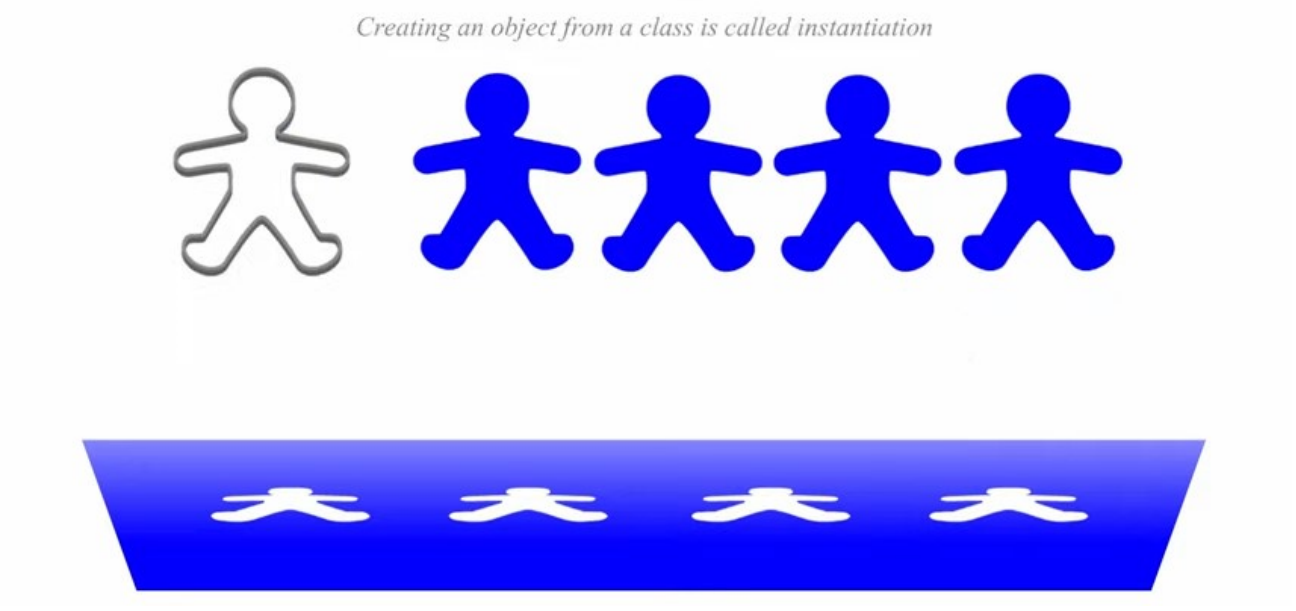
TERMINAL

```
mac-arajendran:Desktop arajendran$ php oop.php
Payment Gateway Name: Stripe
Payment Gateway Name: Paypal
mac-arajendran:Desktop arajendran$
```



2-Vue générale sur la programmation oop :

- La programmation oop est un paradigme de programmation.
- Elle consiste a séparé notre code en classes et objets.
- Elle a plusieurs avantages par rapport a la programmation procédurale.
- Le code est réutilisable.
- Plus simple a maintenir.
- Permet de réaliser des applications extensibles.



3-Les classes:

Les classes sont comme un plan pour créer les objets.

- Une classe est définit par la syntaxe : class ClassName{}
- Dans une classe on définit les propriétés, les méthodes.
- Les propriétés sont les caractéristiques et les variables d'une classe.
- Les méthodes sont les fonctions d'une classe.
- Exemple de mon code :

```php
class User{
    private $username;
    private $password;
    private $name;
    private $email;
    private $phone;
    private $adress;
    private $role;



    function __construct(  $username,$password, $name,$email, $phone, $adress, $role='Client')
    {
        $this->username=$username;
        $this->password=$password=password_hash($password,PASSWORD_DEFAULT);
        $this->name=$name;
        $this->email=$email;
        $this->phone=$phone;
        $this->adress=$adress;
        $this->role=$role;
    }
```

4-Les objets:

- On les créent a partir des classes.
- Pour créer un objet on utilise la syntaxe: new ClassName()

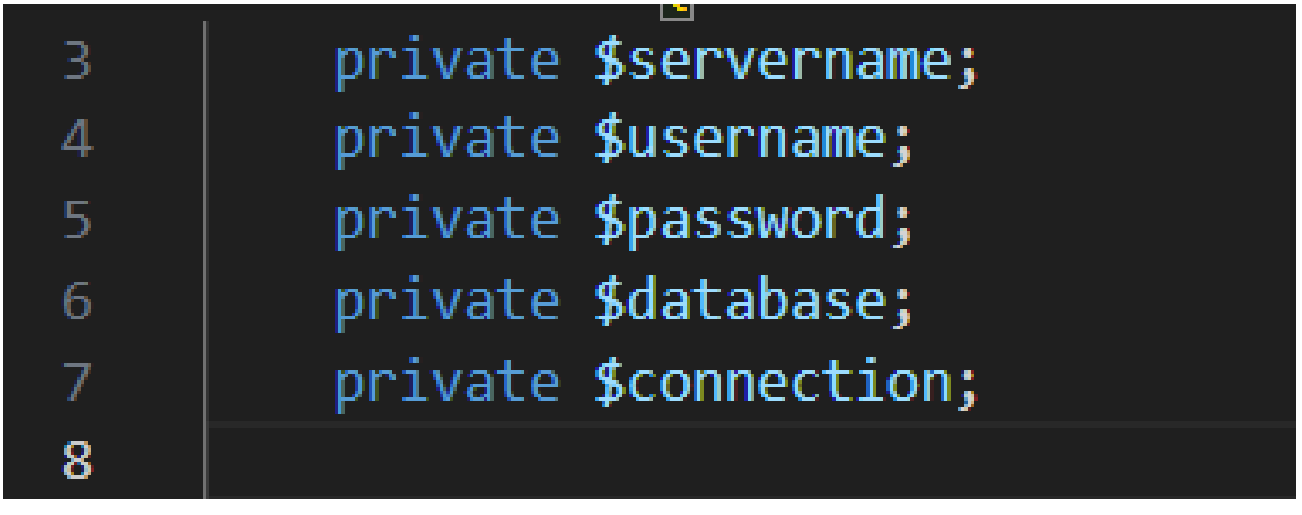
- On parle d’instanciation de classe.
- Exemple d’instanciation :

```php
$user = new User($username, $password, $name, $email, $phone, $adress);
```

5-Visibilité de propriété:

La Visibilité des propriétés détermine comment les propriétés peuvent être accédé:

- Public : peut être accédée depuis l'intérieur ou l'extérieur de la classe.
- Private: ne peut être accédée qu'à l'intérieur de la classe où elle est déclarée.
- Protected: peut être accédée à l'intérieur de la classe où elle est déclarée et les classes qui héritent de celle ci .
- Exemple:



6-Constructor:

- Une méthode qu’on défini dans une classe.
- Elle permet de déterminer les propriétés avec lesquels les objets de la classe seront créer.
- On la définit avec la syntaxe : __construct(paramètres)
- Exemple :

```php
function __construct(  $username,$password, $name,$email, $phone, $adress, $role='Client')
{
    $this->username=$username;
    $this->password=$password=password_hash($password,PASSWORD_DEFAULT);
    $this->name=$name;
    $this->email=$email;
    $this->phone=$phone;
    $this->adress=$adress;
    $this->role=$role;
}
```

7-This:

- Elle fait référence a l'objet actuelle.
- Elle permet d'accéder aux propriétés et méthodes.

8-Get & Set:

- La méthode get permet d'accéder a une propriété.
- La méthode set permet de modifier les propriétés.
- La méthode set prend un paramètre.

9-Destructor:

- Une méthode qu'on défini dans une classe.
- Elle permet de détruire un objet.
- On la définit avec la syntaxe : __destruct

10-Héritage:

- Quand une classe hérite d'une autre il hérite toutes ses méthodes et propriétés.
- On parle de classes descendantes ou sub-classes.
- On utilise la syntaxe class ClassName extends ClassMere {}
- Les classes filles ont accès aux propriétés "protected".

11-Final:

- On peut l'utilisé pour des classes méthodes ou propriétés.
- Permet de prévenir l'héritage ou la modification des classes, méthodes, propriétés.
- Syntaxe : final class ou final function

12-Méthode statique:

- Les méthodes statiques peuvent être appelée sans instancié la classe.
- Les méthodes statiques ne peuvent accéder qu'aux propriétés statiques.
- On définit les classes statiques en utilisant la syntaxe : static function
- On fait appel a une méthode statique avec la syntaxe : NomClasse :: NomMethode()
- Exemple:

```php
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }
}

// Call static method
greeting::welcome();
?>
```